REMARKS

Reconsideration and further examination of this application is respectfully requested. Claims 1, 8, 9, and 16 have been amended to more clearly state the claim limitations and to better place the claims in condition for allowance. Dependent claim 18 has been added to include an additional limitation disclosed in the specification. Dependent claims 17 and 19 have been added to accommodate changes made to claims 1 and 9, respectively. Claims 2 and 5-7 have been amended to reference new claim 17 as the parent claim. Claims 10 and 13-15 have been amended to reference new claim 19 as the parent claim. Claims 3-4 and 11-12 are presented in original form No new subject matter was added by the new claims or the amendments to the claims. All material in the new claims and any amendments to the claims are supported by the specification. With the addition of three new dependent claims, the total number of claims is 19 and the total number of independent claims is 2, thus, no additional fees are required. Therefore, Applicant submits claims 1-19 for further examination.

In the subject Office Action, the Examiner rejected claims 1, 3-9, and 11-16 under 35 U.S.C. 102(e) as being anticipated by Bates et al. (USPN 6,694,036). The Examiner also rejected claims 2 and 10 under 35 U.S.C. 103(a) as being unpatentable over Bates et al. in view of "How Debuggers Work" by J.B. Rosenberg, 1996.

In the subject Office Action, the Examiner asserted that Bates et al. discloses creating an annotated source code. Bates et al. discloses debugger software that operates on the compiled code and not on the high-level computer source code. Bates et al. defines "debugger software" as software "that executes a user's program in a controlled manner" Bates et al. at column 1, lines 39-40. Bates et al. discloses "a method of debugging code, comprising displaying a user interface screen of a <u>debugging program</u>; and in response to an event, displaying the value of the variable and a comment associated with the variable in the user interface screen." Bates et al. at the Abstract. Hence, it is apparent that Bates et al. discloses modifications to the implementation of a debugger program and not the creation of a separate instance of the source code containing additional output statements that is to be compiled and run in place of the original source code. At column 6, lines 24-32 of Bates et al., it is also disclosed that:

"In operation, the compiler 121 parses the source code 118 to produce object code. The object code can then be linked by the linker 122 to produce a complete program 119. <u>As part of the compilation process, the compiler 121 produces a symbol table</u> 120 which is a collection of symbols (variable types and types along with scoping information) within the program 119. <u>A portion of the content of the symbol table 120 makes up debug data used to advantage in embodiments of the invention</u>." (emphasis added).

Therefore, Bates et al. discloses a system that <u>requires the source code to be compiled into object code</u>, and then have the debugger software interact with the compiled object code and the associated symbol table during the execution of the compiled program. At column 10, lines 35-46 (with reference to Fig. 5A) of Bates et al., it is further disclosed that:

"At compile time, the source code 118 is compiled by the compiler 121, which outputs object code. The object code is then linked by the linker 122 to produce the resulting computer program 119 which is subsequently loaded by the loader 117. The program 120 then begins executing. Note that the source code 119 includes a call 508 to the access routine 156_1 (shown in Fig. 1). . . . Following compilation, the call 508 has been replaced with a call 512 to the debugger access routine 156_3."

Thus, Bates et al. discloses modifications to the compiler that permit the compiler to replace function calls in the object code with a "debugger" function call. The code modifications of Bates et al. are modifications to the compiled object code and not modifications to a separate instance of the source code. Bates et al. does not disclose the addition of any statements to the original source code, but only changes to the machine readable object code derived during the compilation process. Since Bates et al. discloses modifications and operations on the machine readable compiled object code, Bates et al. also does not disclose compiling and running a separate annotated source code that is capable of creating a log file to trace program operation.

As currently amended, Applicant's claim 1 recites "creating <u>a separate computer source code</u> to hold an annotated source code." Applicant therefore claims a method that <u>creates a separate instance of computer source code and not the creation of compiled object code</u>. Applicant's separate computer source code contains the original source code plus automatically adding additional output statements that annotate the original source code. The log file may then be used to trace through code operation to find the source of problems that may be difficult to locate using debugger software because the debugger software modifies the compiled object code and/or locating the problem requires extensive operation of the program that may not be conducive to debugger software operation. Bates et al. discloses a system that compiles the computer source code into object code and either makes modifications to the compiled object code or accesses the symbol table resulting from the compilation process. Hence, Bates et al. does not disclose, teach or suggest the creation of a separate computer source code to hold annotated source code as recited in claim 1 of the present invention.

Further, as currently amended, Applicant's claim 1 also recites "reading said portion of said computer source code . . . ; determining if said portion of said computer source code comprises an executable statement; writing said executable statement to said annotated source code . . . ; constructing an output statement . . . ; [and] writing said output statement to said annotated source code." Applicant therefore claims a system that creates a separate annotated source code containing the executable statement of the computer source code plus the newly constructed output statements that write program operation tracing messages to a log file. Bates et al. discloses a system that operates on and makes changes to the machine readable compiled object code, but does not make modifications to either computer source code or a copy thereof. Hence, Bates et al. does not disclose, teach or suggest creating a separate instance of the computer source code that contains the executable statements of the computer source code plus output statements that will write to a log file for tracing program operation as recited in claim 1 of the present invention.

Further, as currently amended, Applicant's claim 1 also recites "causing said annotated source code to be executed in place of said computer source code." Applicant therefore claims a system that executes a separate, annotated, source code in place of the

original computer source code. Bates et al. discloses a system that operates on and makes changes to the machine readable compiled object code, but does not create and execute a separate, annotated source code. Hence, Bates et al. does not disclose, teach or suggest executing a separate annotated source code in place of the original computer source code as recited in claim 1 of the present invention.

In the subject Office Action, the Examiner further asserted that Bates et al. discloses setting a verbosity level to a predetermined level as shown in Fig. 4B at #424. From the description with regard to Figs. 4A-C, Bates et al. discloses "a compilation process during which debug data is gathered." A variety of questions regarding the computer source code being parsed are asked during the compilation process described in Figs. 4A-C of Bates et al. The questions asked concern the type of statement or variable found in the computer source code being parsed as part of the compilation process. For instance, one question is "Comment Found?" Bates et al. at Fig. 4A, #412. The possible responses for each question are "No," in which case the compiler moves on to the next question, or "Yes", in which case the compiler associates debug data with the statement or variable in question. Bates et al. does not disclose making a decision on what level of debug data to include based on a predetermined level or switch.

As currently amended, Applicant's claim 1 recites "setting a verbosity level to a predetermined level; . . . reading said verbosity level to determine desired content for said annotated source code; [and] constructing an output statement comprising said desired content for said executable statement according to said verbosity level." Applicant therefore claims a verbosity level that is used to control the content of output statements to be added to the annotated source code. The content of the output statement determines how verbose, or lengthy, the log file recording program operation will become when the annotated source code is executed. Bates et al. discloses a system that makes decisions at compile time about what debug information to include in the machine readable compiled object code based on the type of statement found, not based on a separate verbosity level that scales the overall amount of information included in an output log file based on a level desired by the user. Hence, Bates et al. does not disclose, teach or suggest using a verbosity level to control the desired content of the output statements for an annotated source code as recited in claim 1 of the present invention.

Further, as currently amended, Applicant's claim 1 recites "constructing an output statement comprising said desired content . . . said output statement further comprising commands that write said desired content to a log file; . . . [and] causing said annotated source code to be executed in place of said computer source code such that said annotated source code operation is recorded in said log file." Applicant therefore claims a method of debugging that causes a log file to be created that records a trace of the operation of the annotated source code. Bates et al. does not disclose a log file. Hence, Bates et al. does not disclose, teach, or suggest the creation of a log file that records the operation of the annotated source code execution as recited in claim 1 of the present invention.
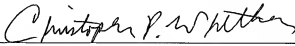
Therefore, Bates et al. does not establish a *prima facie* case under 35 U.S.C. 102(e) according to MPEP §§ 706.02 and 2131 for Applicant's claim 1. "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628 (Fed. Cir. 1987). As shown above, Bates et al. does not disclose, teach or suggest a separate computer source code to hold an annotated source code, adding output statements to the annotated source code that write data to a log file, executing the separate annotated source code in place of the computer source doe, using verbosity level to control the desired content of the output statements, or creating a log file that traces the operation of the annotated source code execution. Consequently, independent claim 1 is not anticipated by Bates et al. Independent claim 9 includes similar limitations as claim 1 and is considered patentable for the same reasons set forth above.

Since independent claims 1 and 9 are not anticipated by Bates et al., dependent claims 3-8 and 17-18 depending from claim 1, and dependent claims 11-16 and 19 depending from claim 9 are also not anticipated by Bates et al. Since Bates et al. does not anticipate the presently claimed invention for independent claims 1 and 9, the combination of Bates et al. with other references, including "How Debuggers Work" by J.B. Rosenberg, to show obviousness of dependent claims 2 and 10 is improper and the Examiner has failed to make a proper *prima facie* case of obviousness as is required under 35 U.S.C. 103(a). Hence, dependent claims 2 and 10 depending from independent claims 1 and 9, respectively, are not rendered obvious by Bates et al. in view of "How

Debuggers Work" by J.B. Rosenberg. Therefore, this application is now considered to be in condition for allowance and such action is earnestly solicited.

Dated this 20th day of December, 2006.

COCHRAN FREUND & YOUNG LLC

By: _____

Christopher P. Whitham   #53,769
Attorney for Applicant
2026 Caribou Drive, Suite 201
Fort Collins, CO 80525
(970) 492-1100
Fax: (970) 492-1101